Spark Assignment

Joseba Hernández Bravo and Jorge Vicente Puig

January 2022

UAB

Universitat Autònoma
de Barcelona

Before starting solving the proposed questions we have to store the data by using the command `val transactions = spark.read.option("inferSchema","true").option("header","true").csv("transactions.csv")`. Also, we will use `show(5)` on all commands (where the results obtained are more than 5 rows), to only show the 5 first rows.

Question 1: ***List of products purchased in Stockholm.***

Firstly, we have to select the column product with the command `select` and then impose the condition of being purchased in Stockholm by the command `where(expr())`.

Command: `transactions.select("product").where(expr("city=’Stockholm’")).show(5)` .

Results:

```
+-------+
|product|
+-------+
|     58|
|     42|
|     74|
|     68|
|     65|
+-------+
```

Question 2: ***Sort the transactions by quantity and get the product id with the largest quantity in a transaction.***

For sorting we will use the command `orderBy()`. The product with the largest quantity will be the first of the table.

Command: `transactions.orderBy(desc("quantity")).show(5)` .

Results:

```
+---+--------+-------+--------+--------+-----+----------+
| id|customer|product|provider|quantity|price|      city|
+---+--------+-------+--------+--------+-----+----------+
|  9|      31|     58|      96|      99|   18| Stockholm|
|565|      18|     59|      18|      50|   97|Parachinar|
|182|      64|     65|      50|      50|   81| Kozlovice|
|232|      58|     41|      88|      50|   27|     Žalec|
|243|      41|     72|      82|      50|   63|   Sanfang|
+---+--------+-------+--------+--------+-----+----------+
```

Therefore, the product with largest quantity in a transaction is the 58.

Question 3: ***Get the list of unique product ids from the dataset.***

The command that will take unique values is `distinct()`. Then, we need to `select` the products column and get the unique values of it.

Command: `transactions.select("product").distinct().orderBy("product").show(5)` .

Results:

```
+-------+
|product|
+-------+
|      1|
|      2|
|      3|
|      4|
|      5|
+-------+
```

Question 4: ***How many products in total and how many different products we have in the input dataset?***

For this operation we will use the `agg` command with the proper operation. We obtain the total products purchased on the data by `sum("quantity")`, the total products that appear on the data set counting repetitions by `count("product")` and the unique products of the data set by `countDistinct("product")`.

Command: `transactions.agg(sum("quantity"),count("product"),countDistinct("product")).show`

Results:

```
+-------------+--------------+--------------+
|sum(quantity)|count(product)|count(product)|
+-------------+--------------+--------------+
|        26085|          1000|           100|
+-------------+--------------+--------------+
```

Therefore, we can see that are 26085 of total product purchased, 1000 products purchased (with repetitions) and 100 different products purchased.

Question 5: ***Count the number of purchases for each city. The result should be a list of cities and number of purchases made.***

The group key are the cities and then an aggregation function has to be used.

It has been decided to use both aggregate functions sum and count due to the interpretation of the number of purchases for each city, where the sum will represent the total number of purchases (the products that has been bought) and the count the total number of purchases (the total number of transactions).

Command: `transactions.groupBy("city").agg(sum("quantity"),count("quantity")).`
`orderBy(desc("count(quantity)")).show(5)` .

Results:

```
+----------------+-------------+---------------+
|            city|sum(quantity)|count(quantity)|
+----------------+-------------+---------------+
|       Stockholm|          260|              8|
|       Guadalupe|           63|              2|
|      Tambakbaya|           17|              2|
|         Adtugan|           41|              2|
|Paris La Défense|          73|             2|
+----------------+------------+--------------+
```

Question 6: ***How many customers have a transaction with a product price between 80 and 100?***

We first filter the data set with the command `where(expr())`, then we can count the customers with an aggregate command using a `countDistinct()` function, taking care of avoiding repetitions.

Command: `transactions.where(expr("price>80")).where(expr("price<100")).`
`agg(countDistinct("customer")).show` .

Results:

```
+---------------+
|count(customer)|
+---------------+
|             88|
+---------------+
```

Question 7: ***Provide a list of cities with the maximum value of products purchased in a transaction. Then sort them by quantity of products and provide the city with the largest quantity.***

We first have to group by cities and then by products. We will use a maximum aggregate function on quantity column in order to know the maximum quantity purchased.

Command: `transactions.groupBy("city","product").agg(max("quantity")).orderBy(desc("max(quantity)"))`
`.show(5)` .

Results:

```
+---------+-------+-------------+
|     city|product|max(quantity)|
+---------+-------+-------------+
|Stockholm|     58|           99|
|Kovylkino|    100|           50|
|     Puwa|     51|           50|
|  Caledon|     52|           50|
|     Lang|      2|           50|
+---------+-------+-------------+
```

Question 8: ***Get all city names from dataset together with its minimum product price.***

We have to group by cities and then use the minimum aggregate function on column price.

Command: `transactions.groupBy("city").agg(min("price")).show(5)` .

Results:

```
+----------+----------+
|      city|min(price)|
+----------+----------+
|   Ilinden|        78|
|    Salamá|        31|
|   Hanover|        23|
|  Izyaslav|        82|
|Siemkowice|        43|
+----------+----------+
```

Question 9: ***Count all the money spent by people in Stockholm. Could you provide a list of all cities and money spent?***

We first have to filter the data set with the condition of city being Stockholm using `where(expr())` command, then using `agg(expr())` we can define an aggregate expression that computes the money spent in that city.

Command: `transactions.where(expr("city='Stockholm'")).agg(expr("sum(quantity*price) as money_spent")).show` .

Results:

```
+-----------+
|money_spent|
+-----------+
|      10294|
+-----------+
```

For a general case is enought to first group by city and then use the same expression as before.

Command: `transactions.groupBy("city").agg(expr("sum(quantity*price) as money_spent")).orderBy(desc("money_spent")).show(5)` .

Results:

```
+------------+-----------+
|        city|money_spent|
+------------+-----------+
|   Stockholm|      10294|
|        Muli|       7046|
|  Salegading|       6584|
|Sumurnanjung|       4850|
|  Parachinar|       4850|
+------------+-----------+
```

Question 10: ***Using the providers.csv dataset, find the names of the providers of the list of cities of question number 5***

Firstly, we have to store the results obtained on exercise 5 with the command `val ex_5 = transactions.groupBy("city","provider").agg(sum("quantity"))` and the dataset of providers with `val providers = spark.read.option("inferSchema","true").option("header","true").csv("providers.csv")`. Then, we can join both datasets with a condition between them that relates the id of providers with the providers on exercise 5.

Command: `ex_5.join(providers).where(providers("id") === ex_5("provider")).show(5)` .

Results:

```
+---------+--------+-------------+---+----------+-----------+
|     city|provider|sum(quantity)| id|  provider|       city|
+---------+--------+-------------+---+----------+-----------+
|     Bāfq|       1|            4|  1|   Dynabox|    Astorga|
|Mojokerto|      22|            7| 22|    Trudoo|  Dongobesh|
|  Pizarro|      88|           42| 88|     Quimm|  København|
|  Xiangfu|      41|           35| 41|    Oyondu|    Maniowy|
|    Kaiaf|      81|           39| 81|Shuffletag|Sindangsari|
+---------+--------+-------------+---+----------+-----------+
```